VIDEO BASIC

20 LECCIONES DE BASIC
PARA APRENDER CON EL SPECTRUM

INGELEK



JACKSON

- Joystick y paddle
- Información analógico-digital
- Cómo controlar un joystick - los interfaces
- LEN
- VAL
- TO IN OUT
- Las operaciones sobre cadenas - "slicing"
- Videoejercicios
- Videojuego N.º 9

9

Spectrum
16K/48K/PLUS

VIDEO BASIC
Una publicación de
INGELEK JACKSON
Director editor por INGELEK:
Antonio M. Ferrer
Director editor por JACKSON HISPANIA:
Lorenzo Bertagnolio
Director de producción:
Vicente Robles
Autor: Softidea
Redacción software italiano:
Francesço Franceschini,
Stefano Cremonesi
Redacción software castellano:
Fernando López, Antonio Carvajal,
Alberto Caffarato, Pilar Manzanera
Diseño gráfico:
Studio Nuovaidea
Ilustraciones:
Cinzia Ferrari, Silvano Scolari,
Equipo Galata
Ediciones INGELEK, S. A.
Dirección, redacción y administración,
números atrasados y suscripciones:
Avda. Alfonso XIII, 141
28016 Madrid. Tel. 2505820
Fotocomposición: Espacio y Punto, S. A.
Imprime: Gráficas Reunidas, S. A.

ISBN del tomo 3: 84-85831-19-5
ISBN del fascículo: 84-85831-11-X
ISBN de la obra completa: 84-85831-10-1
Depósito Legal: M-15076-1985
Plan general de la obra:
20 fascículos y 20 casetes, de aparición quincenal,
coleccionables en 5 estuches.
Distribución en España:
COEDIS, S. A.
Valencia, 245. 08007 Barcelona.
INGELEK JACKSON garantiza la publicación de todos
los fascículos y casetes que componen esta obra y el
suministro de cualquier número atrasado o estuche
mientras dure la publicación y hasta un año después de
terminada.
El editor se reserva el derecho de modificar
el precio de venta del fascículo,
en el transcurso de la obra, si las circunstancias del
mercado así lo exigen.
Agosto, 1985
Impreso en España.



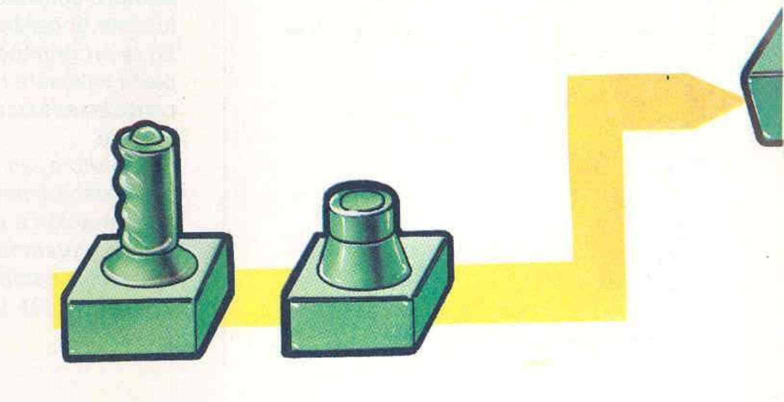
SUMARIO

Introducción

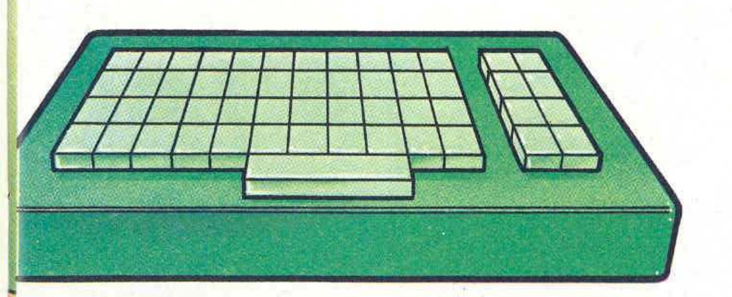
¿Quién no conoce el joystick, esa emocionante palanca con un pulsador, omnipresente en los videojuegos caseros y en los de los bares? ¿Cuántas batallas se han ganado o perdido empuñándolo contra terribles enemigos? Más allá del juego, el joystick, como sus hermanos los paddles, es un periférico de entrada, con su lógica digital y sus principios de funcionamiento que siempre conviene conocer. Esto en lo tocante al hardware.
En la programación añadimos otra pieza valiosa a nuestro mosaico de conocimientos: las funciones de cadena.
Hasta ahora, en el uso de cadenas, nos hemos limitado a su asignación, comparación e impresión. En esta lección veremos como se pueden manipular datos alfanuméricos gracias a LEN, VAL, STR$, TO, IN/OUT.

Paddle y Joystick

Ya hemos visto en las anteriores lecciones que el principal dispositivo de un ordenador, para entrada de datos e informaciones, está constituido por el teclado. Y tanto es así que prácticamente no existen ordenadores – exceptuando aquéllos construidos para aplicaciones muy especiales– que no dispongan de él y que recurran a otros medios para la entrada de datos.
Sin embargo, principal no significa único: hay un montón de periféricos y accesorios disponibles, que en cinscunstancias particulares, pueden ser mucho más útiles y apropiados que el teclado, al igual que le ocurre al hombre cuando desea algo más cómodo y rápido para escribir que la sencilla pero siempre indispensable pluma.
Hoy hablaremos precisamente de estos dos dispositivos: los joysticks y los paddles.
Es muy probable que no te resulte completamente desconocido, dado que sus nombres se emplean con frecuencia entre los apasionados jugadores de los inevitables videojuegos, ya que se usan como principales armas de ataque y defensa contra las pléyades de invasores espaciales.
Y también, cualquiera que haya entrado en un bar no puede no identificar un joystick, reconociendo inmediatamente bajo tal nombre la apasionante palanquita necesaria para escapar de los

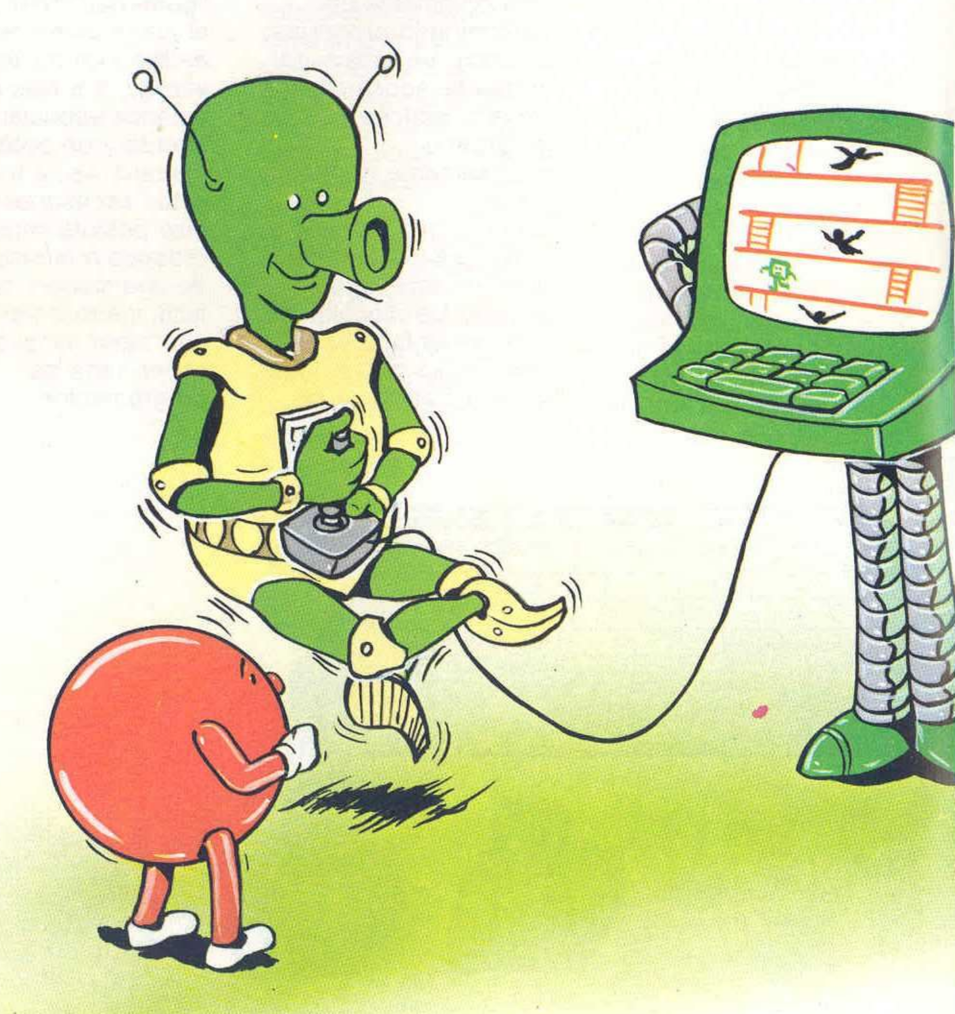


peligros del «bombardeo» enemigo. Será más fácil en cambio, que no te sean demasiado conocidos sus principios de construcción y funcionamiento: el objetivo de esta lección será el de explicarte sus secretos y sus diferencias.
Veamos en primer lugar cuáles han sido los motivos que han determinado su nacimiento y, especialmente en los últimos tiempos, su difusión.
Todo empezó hace pocos años, precisamente con los citados videojuegos: un videojuego no es más que un ordenador especial programado para ejecutar única y exclusivamente un determinado programa; es decir, un ordenador al que se acostumbra a llamar "dedicado". Y su programa, precisamente, es el juego.
No hizo falta mucho para darse cuenta de que era necesario superar las dificultades que presentaba el teclado: en primer lugar, la mayor parte de las teclas hubiera resultado absolutamente inútil para el uso al que estaba destinado el ordenador, y además sería lógico suponer que a los usuarios les resultara más sencillo, y sobre todo mucho más divertido, tener la impresión de "gobernar" directamente el juego como ocurre en las astronaves de verdad, a través de una palanca especial de mando y un botón de disparo. Así, a través de estos accesorios se hizo posible impartir las órdenes a la máquina de una manera rápida y fácil, y sobre todo, sin que fuera necesario saber nada de programación.



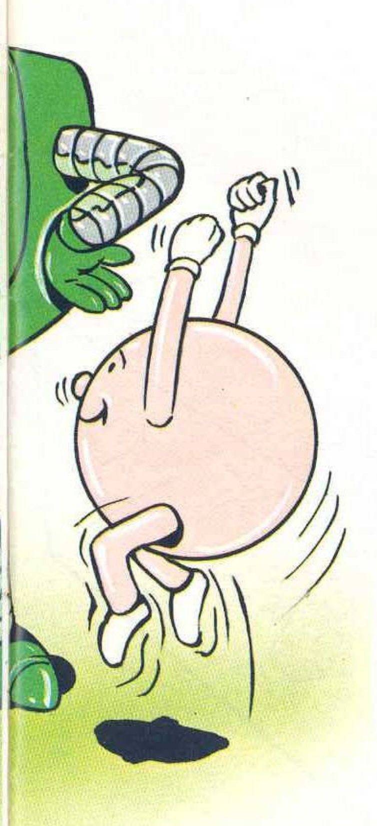
Cómo controlar un joystick

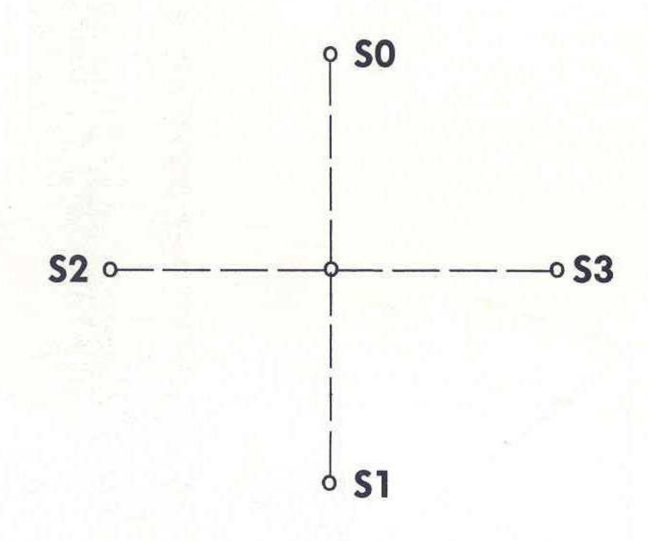
Para una persona no es demasiado natural pulsar botones para ordenar movimientos: si tuviéramos que conducir un coche mediante teclas o pulsadores, no obtendríamos la percepción continua de movimiento que se consigue girando un volante.
Fue así como nació la idea del joystick y, a continuación, la de los paddles (que como luego veremos, son una especie de joysticks más refinados).
Actualmente, el uso de estos dispositivos ya no



es una prerrogativa exclusiva de los videojuegos: casi todos los ordenadores personales permiten su uso a través de algunas conexiones expresamente realizadas en el ordenador mismo.
Por lo tanto, un joystick es –como los teclados– un dispositivo de entrada; su apariencia es la de una palanca que con su movimiento permite desplazar un objeto en pantalla.

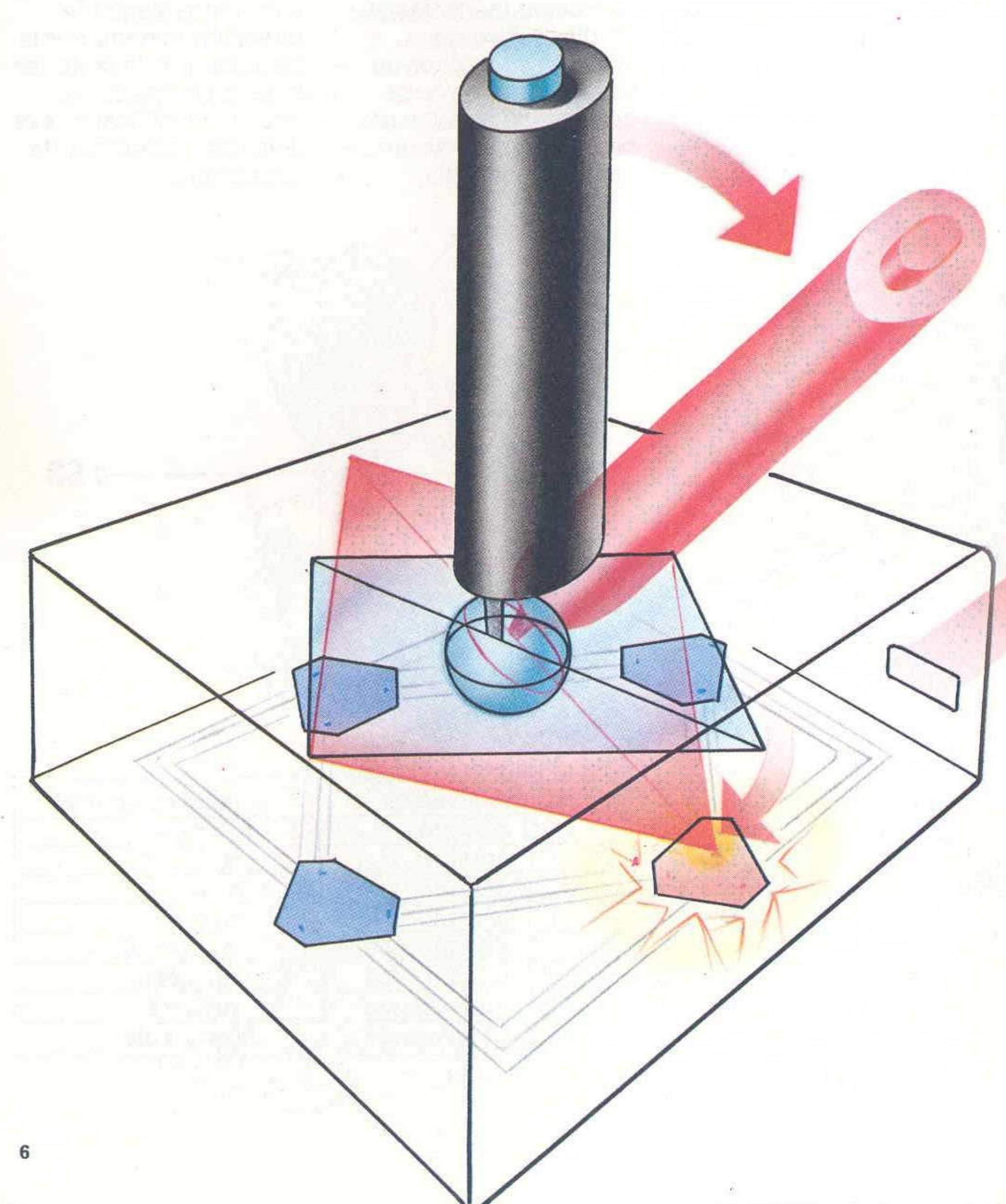
Veamos cómo lo consigue.
En el interior del joystick existen cuatro interruptores que son archivados según la dirección tomada por la palanca: por lo tanto, en base a su estado es posible identificar nueve distintas posiciones de la palanca:





todos abiertos	palanca vertical
SO cerrado	norte
S1 cerrado	sur
S2 cerrado	oeste
S3 cerrado	este
S0, S1 cerrados	norte-oeste
S1, S2 cerrados	sur-oeste
S2, S3 cerrados	sur-este
S3, S0 cerrados	norte-este

A cada uno de estos cuatro interruptores se le asocia un bit en una localización de memoria en el interior del ordenador: bit 1 para interruptor encendido, o bit 0 para interruptor apagado. Un quinto interruptor funciona como pulsador de disparo o como señal de que la posición deseada ha sido alcanzada.

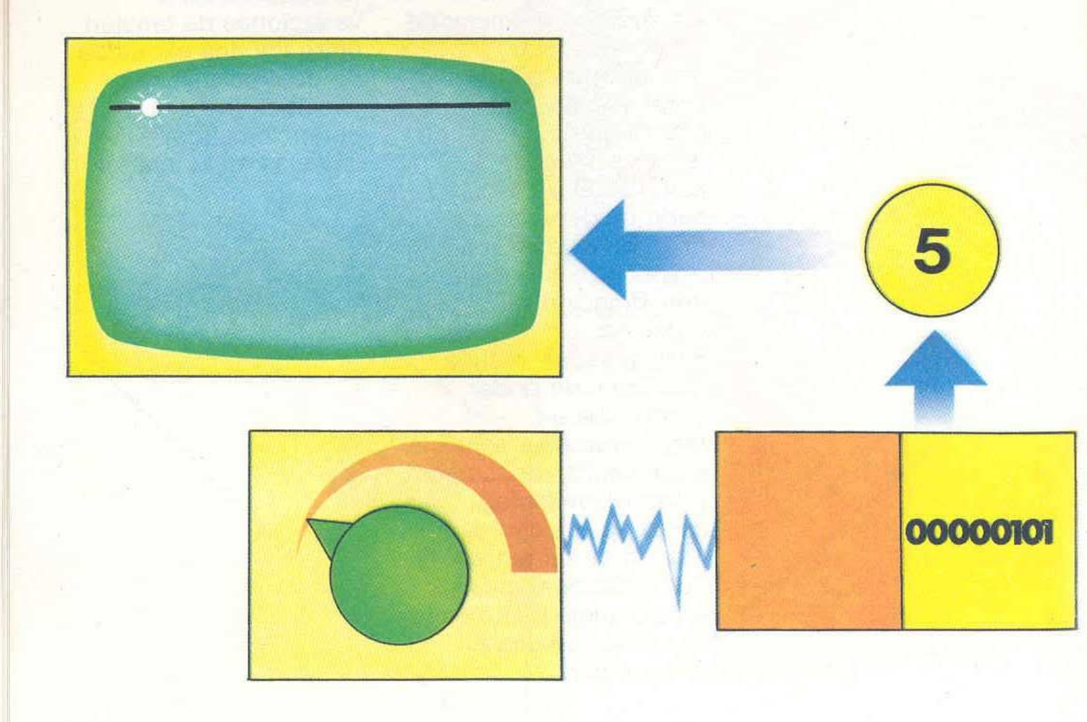


Entonces, a través del programa, será suficiente con leer en su localización de memoria el valor de cada uno de estos bits, para averiguar la posición de la palanca y saber en consecuencia cuál es la acción a realizar (desplazamiento del cursor, de la astronave o del objeto en pantalla).

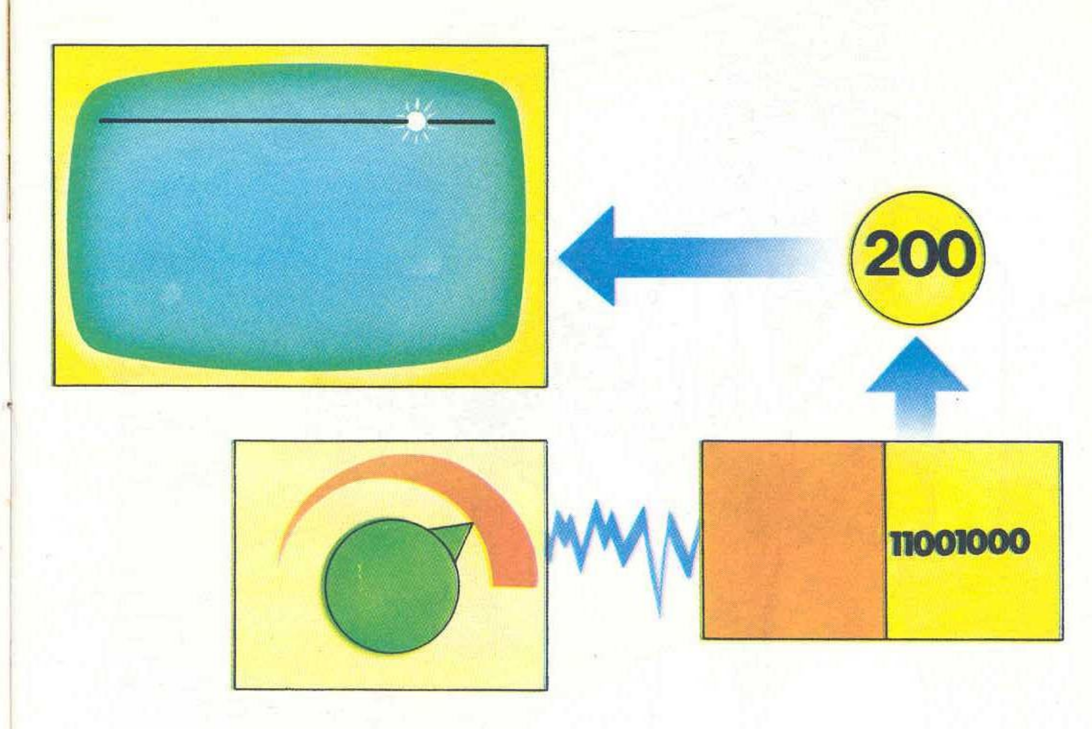
Informaciones analógicas y digitales: interface

El joystick es un dispositivo digital: el dato contenido en la localización de memoria se corresponde exactamente con la combinación de interruptores debida al accionamiento de la palanca. El estado de cada bit depende del estado del interruptor respectivo, sin ninguna aproximación ni redondeo.
Pero, a veces, resulta más cómodo poder contar con un dispositivo más sensible a pequeñas variaciones y desplazamientos, como ocurre con el volante de un automóvil. Por lo tanto, al joystick propiamente dicho se le ha unido el llamado joystick con potenciómetro (paddle). A los efectos de su funcionamiento, un paddle es muy semejante a un joystick: pero en lugar de una palanca, en el paddle aparece una manivela cuya regulación es mucho más ajustada y precisa. Y además el efecto se obtiene de una manera completamente distinta a como ocurría en el joystick: el desplazamiento de la manivela ya no provoca el encendido de interruptores y pulsadores, sino variaciones de tensión entre los dos extremos de dos resistencias variables (o potenciómetros) situadas en el interior

del paddle. El resultado final es un dato que varía constantemente, o como se acostumbra a decir, está constituido por una información analógica. La adecuada conversión de esta información de analógica a digital devuelve bajo la forma de un número de 8 bits la acción especificada por el desplazamiento de la manivela, memorizándola en una determinada localización de la memoria. Como es natural, para efectuar esta conversión, se emplea el interface adecuado, situado entre la unidad central y el paddle, cuyo objeto es transformar la señal continua proporcionada por el paddle, en una información binaria. Mediante la operación de leer la localización

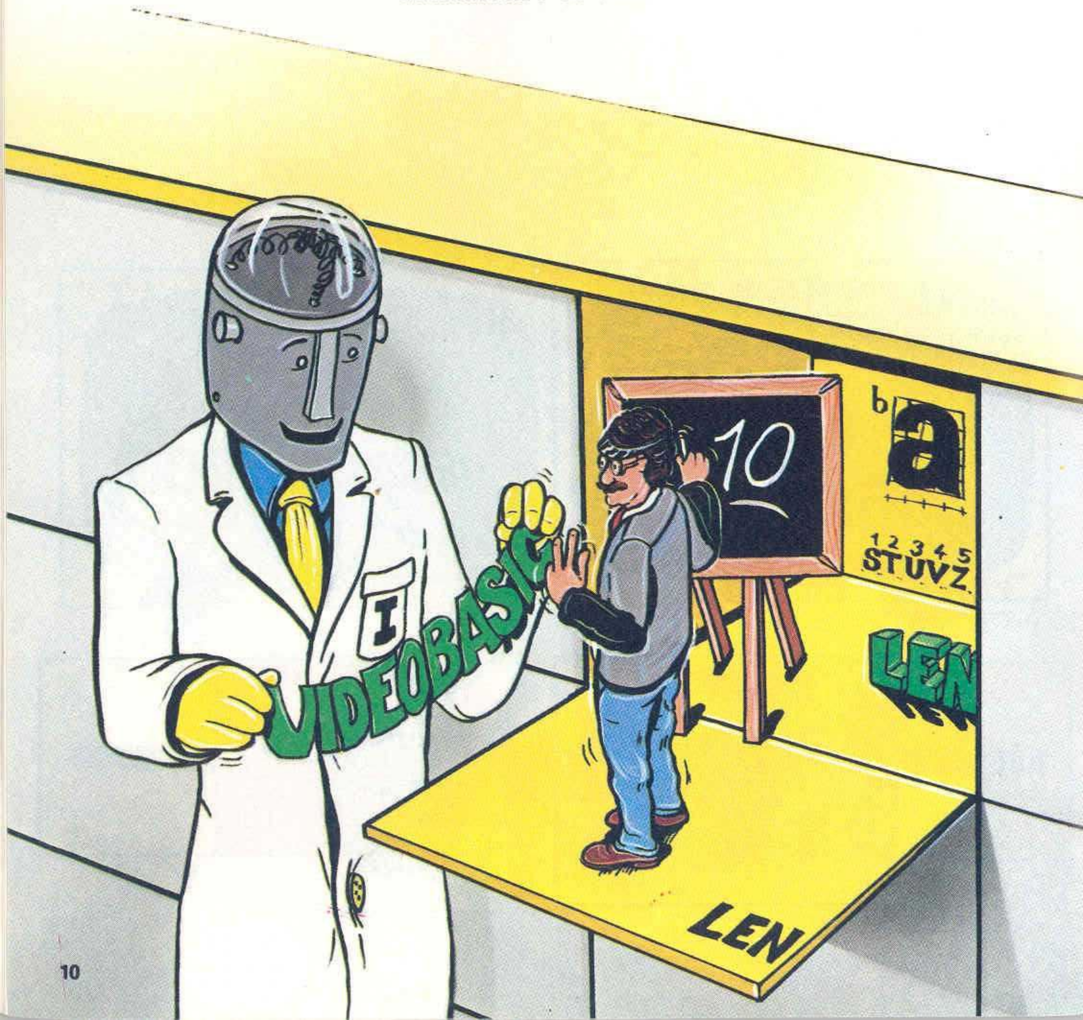


de memoria será posible, como en el joystick, recuperar el movimiento deseado.
Naturalmente, también un paddle tiene su límite de sensibilidad, debido a la operación de conversión; pero de cualquier forma éste queda muy por encima del límite del joystick, permitiendo un total de 256 posibles combinaciones.
Pero esto no quiere decir que un paddle sea forzosamente mejor que un joystick; todo depende del uso y de la sensibilidad necesaria para cada aplicación determinada. Lo que es sin duda más sencillo es conectar el joystick al ordenador, puesto que no se requiere ningún dispositivo de conversión, lo que no ocurre con el paddle.



Los operadores de cadena

Hasta este momento nuestro uso de las cadenas se ha limitado a la asignación, a la comparación, y a la impresión de este tipo de expresiones. En cambio, en muchos programas para ordenadores aparecen extensas manipulaciones y tratamientos de datos alfanuméricos. En esta importante lección intentaremos aprender las instrucciones y comandos que nos permitan efectuar este tipo de operaciones.



LEN

La función LEN (abreviatura inglesa de LENght, longitud) es muy fácil de entender: permite encontrar el número de caracteres que componen una cadena.
Por ejemplo, si deseamos determinar el número de caracteres que componen la cadena "JUAN" será suficiente con indicar:

```
PRINT LEN "JUAN"
```

y en la pantalla aparecerá 4.
La cadena "JUAN" de la que deseábamos contar los caracteres es el argumento de LEN y, por lo tanto, se coloca entre paréntesis inmediatamente después de la palabra LEN.
LEN devuelve un resultado numérico a partir de un argumento que ha de ser forzosamente de tipo cadena, sea variable o constante. Obseva que las comillas no se toman en consideración (lo que es lógico) como partes constituyentes de la cadena.
Es una instrucción muy potente y de uso muy frecuente. Los casos en los cuales puede interesar conocer la longitud de una cadena son numerosísimos; citaremos como ejemplos los encolumnados, los paginados y las comparaciones entre cadenas.

Ejemplos

```
PRINT LEN "CUENTA CORRIENTE"
```

Imprime el número 16.

```
LET A = LEN ("")
```

A la variable se le asignará el valor 0 (la cadena nula no contiene ningún carácter).

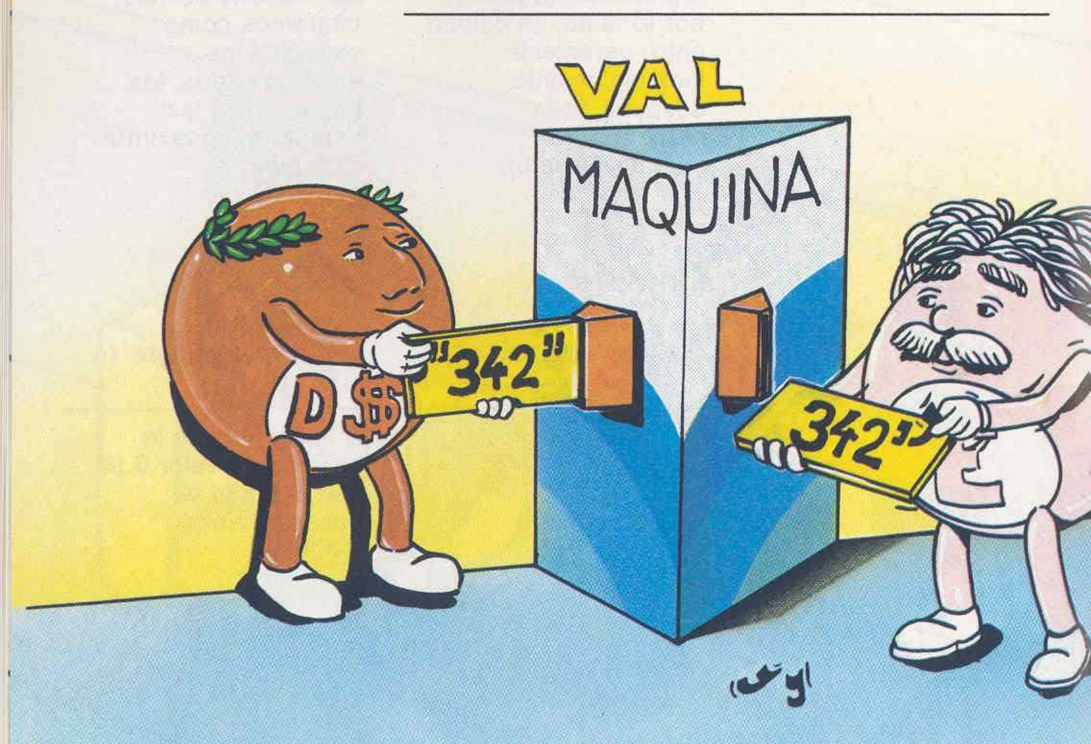
```
LET A$ = "ANA"
PRINT LEN (A$ + "MARIA")
```

```
LET A$ = "ANA"
PRINT LEN (A$) + LEN ("MARIA")
```

Estas dos parejas de instrucciones tienen el mismo resultado, 8. En el primer caso se efectúa en primer lugar la concatenación de las dos cadenas y a continuación el cálculo de la longitud, mientras que en el segundo caso se calculan por separado las dos longitudes para luego sumarlas.

Sintaxis de la función

LEN (cadena)



VAL

Existen algunos casos en los que resulta muy cómodo poder convertir una cadena de números (como "3149") en un valor numérico, para poderlo usar dentro de expresiones numéricas.

La función VAL (abreviatura de VALue, valor) convierte las cadenas en números. Así:

```
LET A = VAL "3149"
```

le asigna a la variable numérica A el valor 3149.
Pero de cualquier manera no debemos malinterpretar esta función: no es capaz de hacer proezas. Una cadena como "TAPON" nunca podrá tomar ningún valor numérico; cuando el intérprete BASIC encuentra una instrucción como:

```
PRINT VAL "TAPON"
```

comprueba si existe una variable con ese nombre.
Si no la encuentra envía el mensaje:

VARIABLE NOT FOUND

pero si la variable hubiera sido asignada anteriormente, hubiera impreso su valor.

Ejemplos

`LET X = VAL "8"`	Asigna a la variable X el valor numérico 8.
`PRINT VAL "54C3"`	Provocará el mensaje de error NONSENSE IN BASIC, puesto que la cadena contiene un carácter no numérico.
`LET P$ = "103"` `LET Q$ = "2"` `PRINT P$ + Q$`	Imprime la suma (es decir, la concatenación) de las cadenas P$ y Q$. Por lo tanto, el resultado será la cadena "1032".
`PRINT VAL P$ + VAL Q$`	Imprime la suma de los valores numéricos 103 y 2, es decir, 105.

Sintaxis de la función

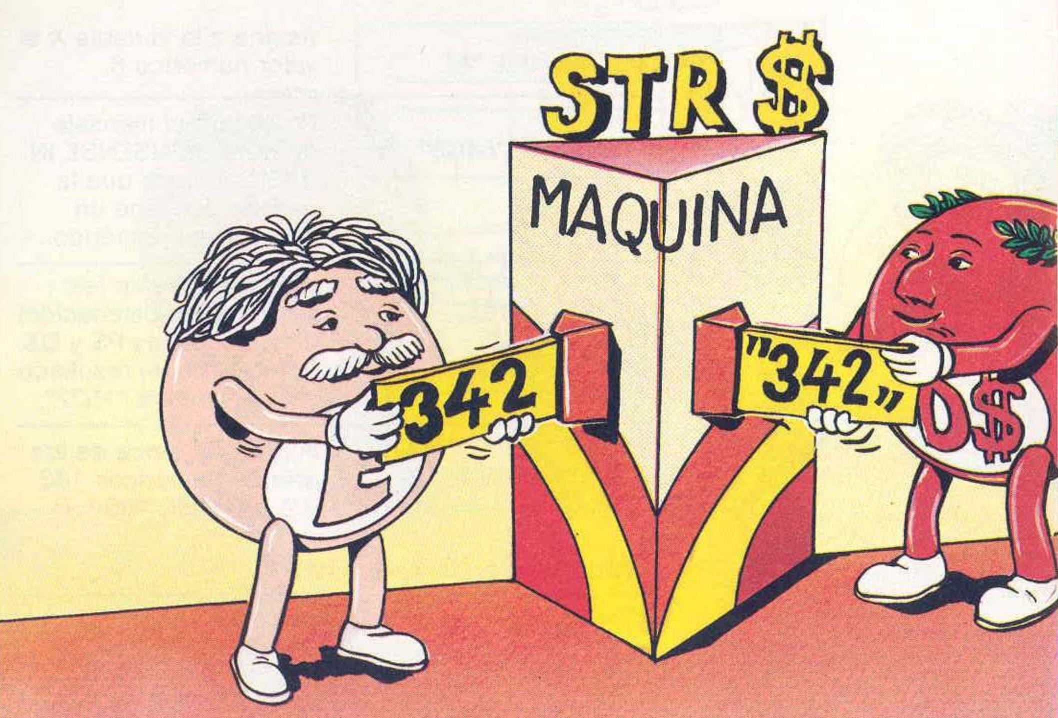
VAL (cadena)

STR$

STR$ (abreviatura de STRing, cadena) es la función recíproca de VAL. Permite transformar una cantidad numérica en una cadena de caracteres.
Cuando deseemos transformar un número (como el 8) en una cadena, de forma que sea posible combinarlo con otras cadenas (por ejemplo "mil") STR$ será la función que necesitemos.
La instrucción

```
PRINT STR$ 8 + " " + "MIL"
```

producirá como resultado la impresión de la cadena "8 MIL".



Ejemplos

LET A$ = STR$ 50	Asigna a A$ el valor alfanumérico "50".
PRINT STR$ (3 * 7 – 2)	Resultado: 19.
PRINT VAL STR$ 50	Es una instrucción inútil, aunque perfectamente legal. El valor numérico 50 se convierte primero en cadena (con STR$) y después se transforma en número. Aparecerá entonces en pantalla 50, entendido como número y no como cadena.

Sintaxis de la función

STR$ (expresión)

TO

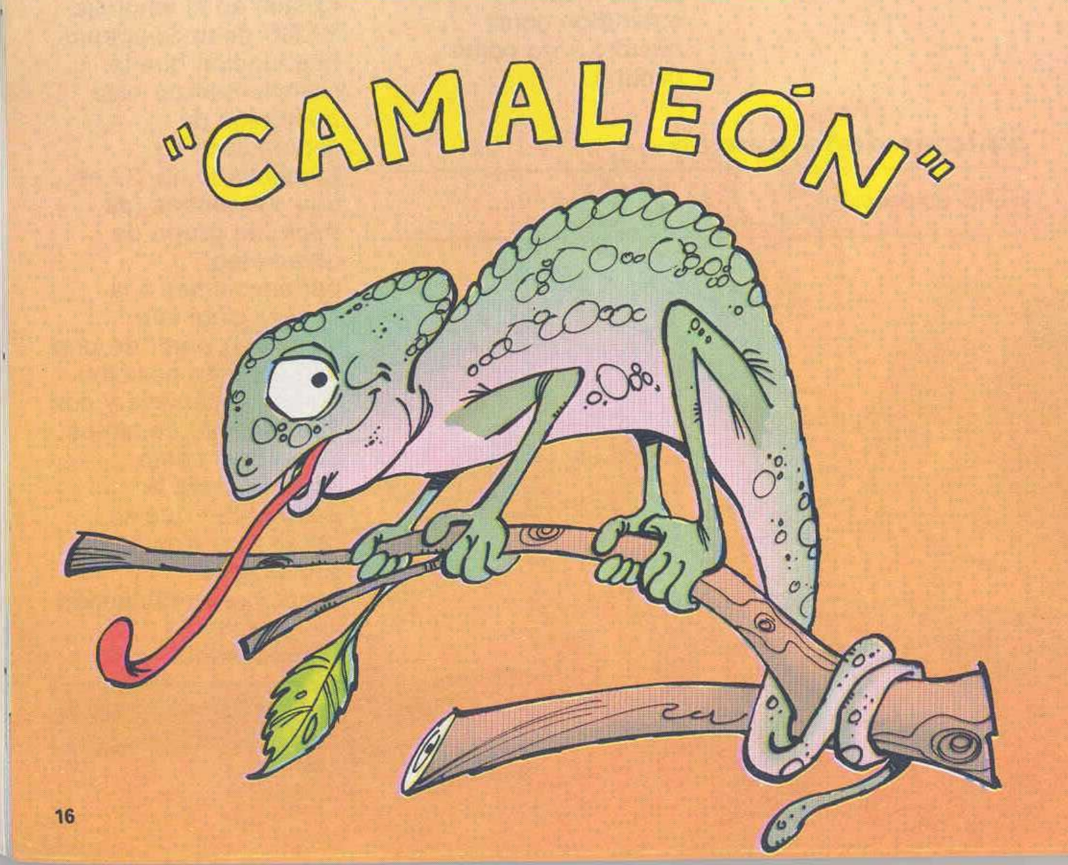
La operación de concatenación permite añadirle caracteres a una cadena (teniendo en cuenta que nunca debe superarse el límite máximo de 255 caracteres) efectuando una especie de suma de cadenas.
En algunos casos, en lugar de añadir, podría ser necesario tener que extraer caracteres de una cadena: para esto existen en el lenguaje BASIC de tu Spectrum una función que te permite realizar esta operación de separación: TO.
El resultado de TO es una subcadena (es decir, un grupo de caracteres pertenecientes a la cadena original) extraída a partir de una determinada posición que se especifica y con una longitud de tantos caracteres como existan, hasta la posición en que se desee que termine la subcadena.
Veamos a continuación un ejemplo esclarecedor:

```
PRINT "JUAN CARLOS" (1 TO 4)
```

Imprime 4 caracteres de la cadena "JUAN CARLOS", partiendo desde la primera posición y terminando en la cuarta. El resultado será la visualización de JUAN. El primer número del interior del paréntesis especifica la posición del carácter desde el que empezará la extracción de la subcadena. El segundo número identifica la posición del carácter en el que debe finalizar la extracción.
Si ahora deseáramos ejecutar la misma operación, pero desde la parte opuesta de la cadena (es decir, si deseáramos separar los últimos cinco caracteres) sería, suficiente indicar:

```
PRINT "JUAN CARLOS" (6 TO 11)
```

y el resultado sería la aparición en pantalla de CARLOS.
Por lo general, el troceado ("slicing") de una cadena, es decir, la subdivisión de una cadena en una o más subcadenas, se efectúa de la siguiente forma: cadena (posición del primer carácter TO posición del último carácter); en la que los

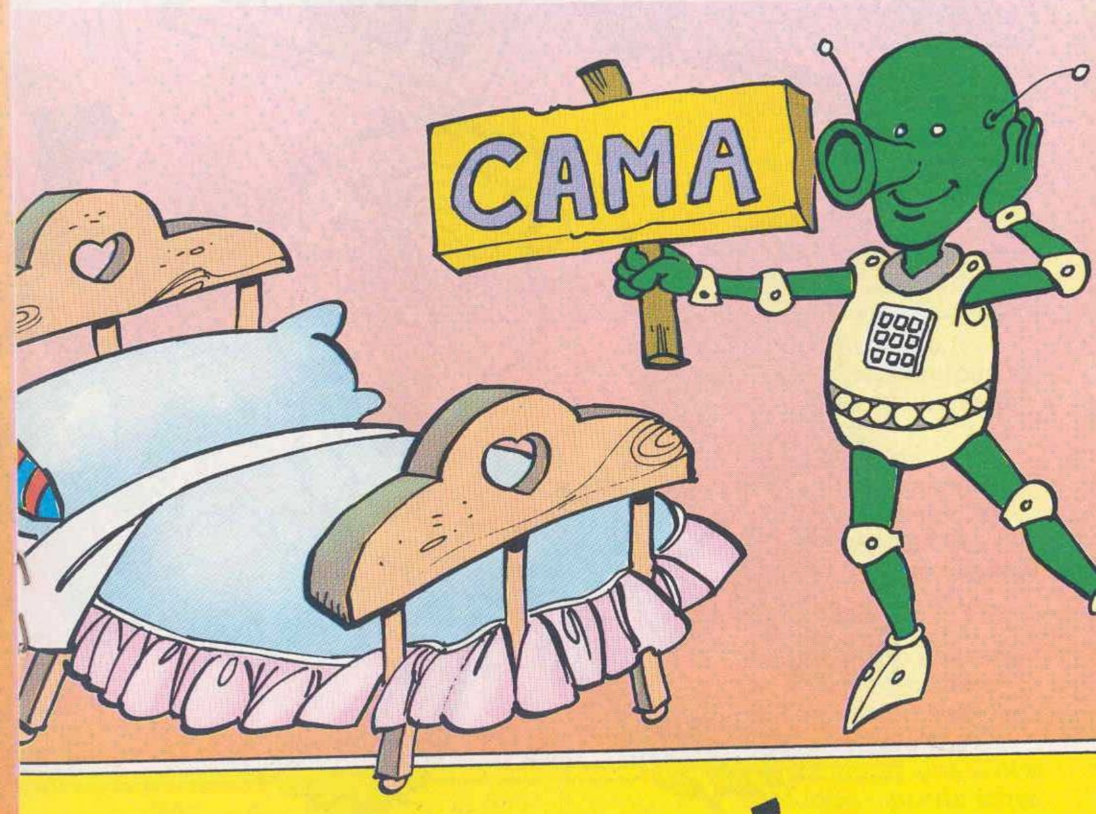


números de las posiciones del primer y del último carácter especifican donde debe empezar y terminar la subdivisión.
De cualquier forma debido al gran número de veces en las cuales la operación de troceado afecta únicamente a un determinado número de caracteres, al principio o al final de una cadena, la casa Sinclair ha hecho posible usar formas abreviadas para simplificar la escritura de esta instrucción. Así:

```
LET A$ = "RADIOAFICIONADO" (1 TO 5)
```

es equivalente a

```
LET A$ = "RADIOAFICIONADO" (TO 5)
```



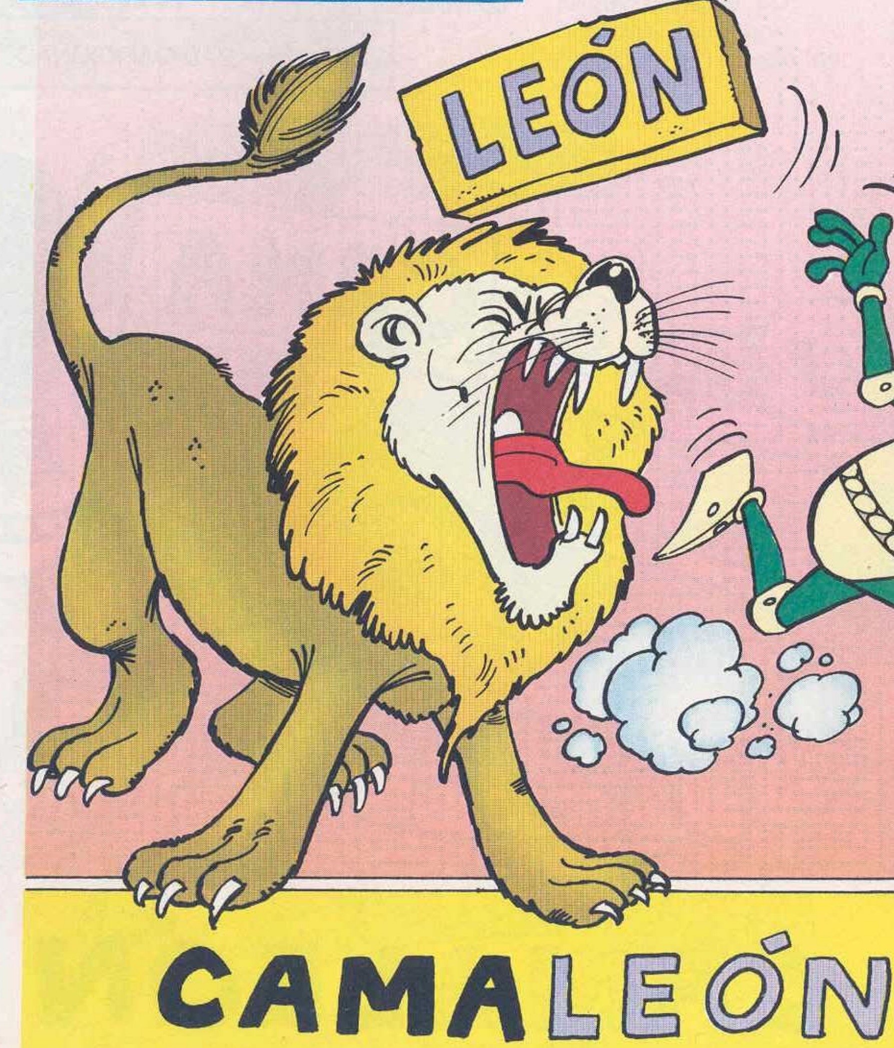
En ambos casos la subdivisión empezará en el primer carácter para terminar en el quinto, asignándole así a la variable A$ la subcadena "RADIO". Para asignarle en cambio a A$ la subcadena "AFICIONADO" hubieramos podido escribir:

```
LET A$ = "RADIOAFICIONADO" (6 TO 15)
```

o bien

```
LET A$ = "RADIOAFICIONADO" (6 TO)
```

Nuesto Spectrum hubiera entendido lo mismo en ambos casos: extraer los últimos 10 caracteres de la cadena inicial.
Si ahora deseáramos obtener como subcadena un solo carácter (por ejemplo, la R de MADRID), tendriamos que escribir:



```
PRINT "MADRID" (4 TO 4)
```

o también:

```
PRINT "MADRID" (4)
```

Y el cuarto carácter de MADRID, la R, se visualizará en pantalla. De cualquier forma, siempre tienes que evaluar muy cuidadosamente los números que marcan la posición de los caracteres del principio y el final de la extracción: valores equivocados o no deseados pueden llevar a subdivisiones absurdas o, peor todavía, provocar mensajes de error. Por ejemplo:

```
LET F$ = "EDICIONES INGELEK" (9 TO 30)
```

es una instrucción equivocada, puesto que la cadena contiene un total de 17 caracteres incluyendo el espacio. También

```
LET F$ = "EDICIONES INGELEK" (–1 TO 6)
```

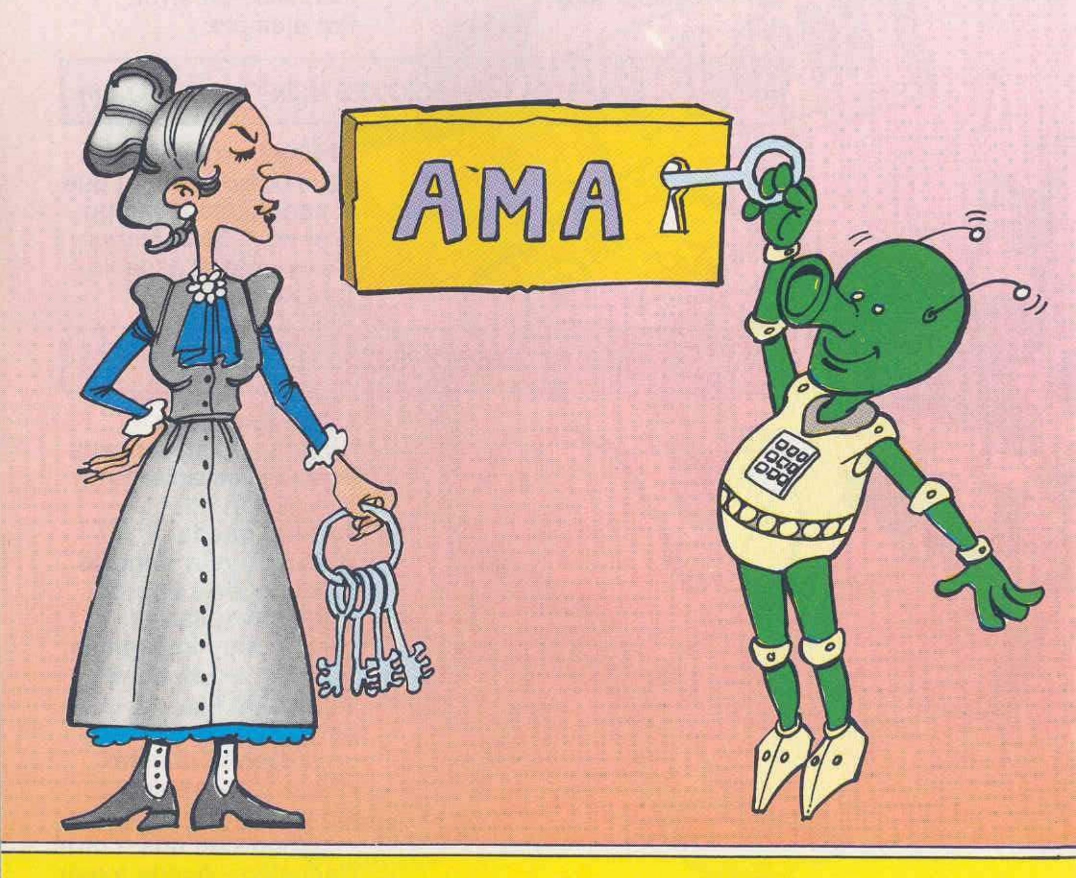


está equivocado. Ninguna palabra puede contener caracteres en la posición –1. Aunque pueda parecerte muy sencillo evitarlos, este tipo de errores resultan en cambio muy frecuentes; muchas veces, en el interior de los paréntesis no aparece un auténtico número, sino una expresión numérica, que —pasando de una a otra ejecución— puede tomar distintos valores. De no haber tomado en

consideración todos los casos posibles, puede ocurrirnos lo que ya hemos comentado, provocando la aparición del fatídico mensaje de error:

INTEGER OUT OF RANGE

El uso más típico de este comando lo tenemos en aquellos programas que piden, como entrada una fecha escrita en la forma DD/MM/AA (como 27/10/60). Con TO resulta muy fácil descomponer la fecha en sus elementos DD/MM/AA.



Veamos como se puede hacer esto:

```
10 INPUT "ESCRIBE LA FECHA (DD/MM/AA)"; F$
20 D$ = F$ (1 TO 2)
30 M$ = F$ (4 TO 5)
40 A$ = F$ (7 TO 8)
50 PRINT D$; " "; M$; " "; A$
```

Al final de este programa, D$, M$ y A$ contendrán respectivamente el día, mes y año de la fecha asignada inicialmente a F$.

Ejemplos

```
LET A$ = "enerofebreromarzo"
PRINT A$ (1 TO 5)
PRINT AS (TO 5)
```

Imprime "enero"

```
LET A$ = "enerofebreromarzo"
PRINT A$ (13 TO 17)
PRINT A$ (13 TO)
```

Imprime "marzo"

```
LET A$ = "enerofebreromarzo"
PRINT A$ (6 TO 12)
```

Imprime "febrero"

```
PRINT "CINTA (TO LEN ("CINTA") –1)
```

El número de caracteres a extraer a partir de la izquierda será 4 puesto que LEN ("CINTA") es 5, y al restarle 1, da 4 como resultado.

```
PRINT "NOCHE" (1 TO 5)
```

Puesto que 5 es igual a la longitud total de la cadena, se imprime la cadena completa "NOCHE".

```
LET A$ = A$ (–1)
```

¡Error! No es posible extraer nada partiendo del carácter –1 de una cadena.

El breve programa siguiente (encontrarás otros en la parte de la lección dedicada a la programación) te muestra una sencilla aplicación de los conceptos explicados hasta el momento.

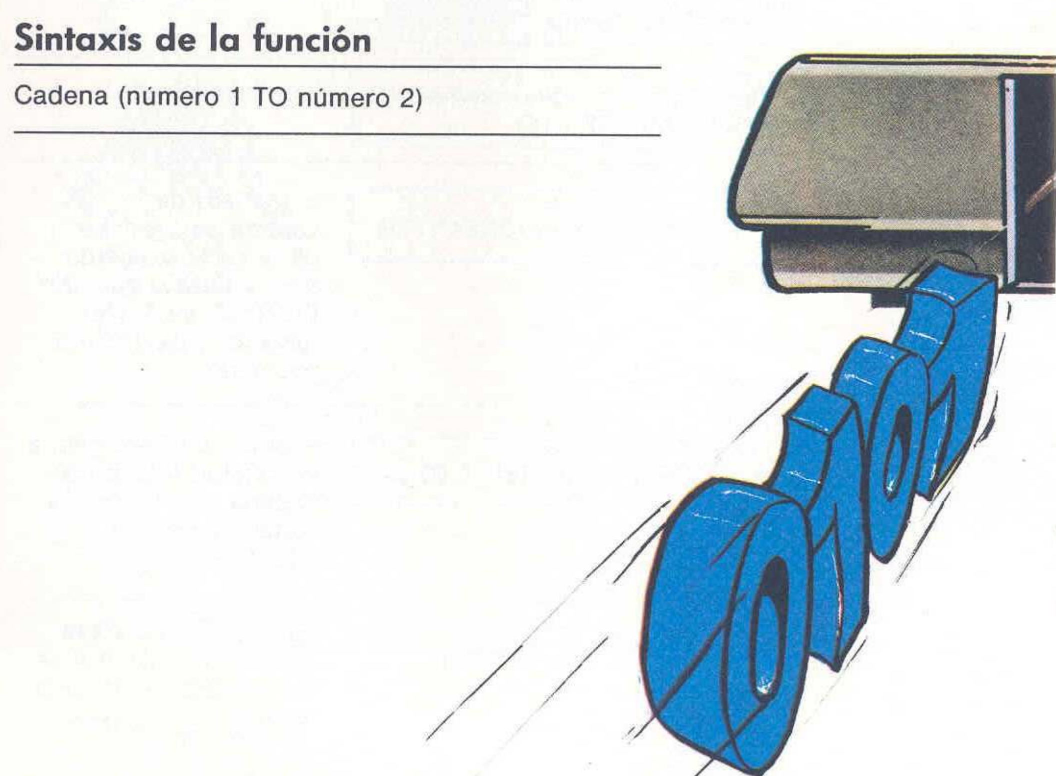
```
10 INPUT "ESCRIBE UNA PALABRA
   CUALQUIERA"; P$
20 FOR I = 1 TO LEN (P$)
30 PRINT P$ (1 TO I); " "; P$ (LEN(P$) – I + 1 TO
   LEN (P$)
40 NEXT I
50 STOP
```

Sintaxis de la función

Cadena (número 1 TO número 2)

IN/OUT

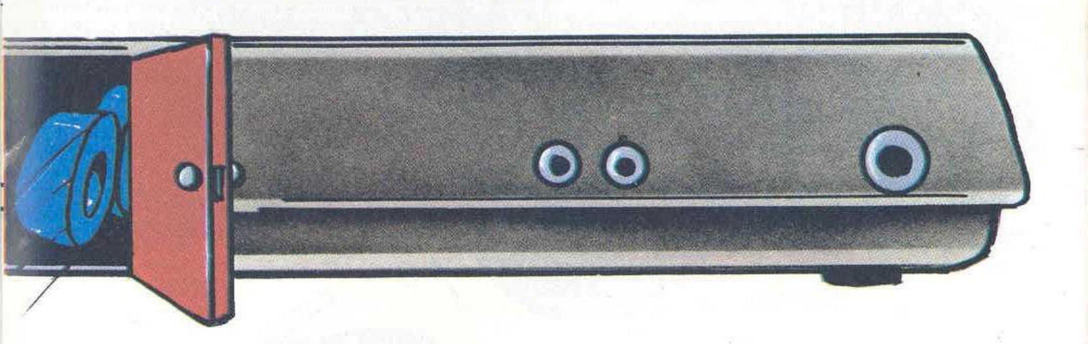
Ya has estudiado que el microprocesador que contiene tu Spectrum es capaz de conectarse a través de códigos especiales y combinaciones con cualquier parte de la memoria: es decir, puede acceder (leyendo



y escribiendo) a todas las localizaciones que componen la memoria. Es por esto, que el BASIC te permite —aprovechándose precisamente de estas capacidades— "curiosear" en las distintas localizaciones a las dos instrucciones PEEK y POKE.

De la misma manera, es posible considerar a las diversas posiciones de memoria no ya como simples almacenes e informaciones, sino como auténticas puertas de comunicación entre el ordenador y el mundo exterior (que según los casos, puede ser el teclado, la impresora, un joystick o la grabadora).

Por lo tanto, el Spectrum proporciona dos instrucciones capaces de acceder a cada dispositivo, de entrada/salida, exactamente igual que permite —a través de PEEK y POKE— referirse a localizaciones



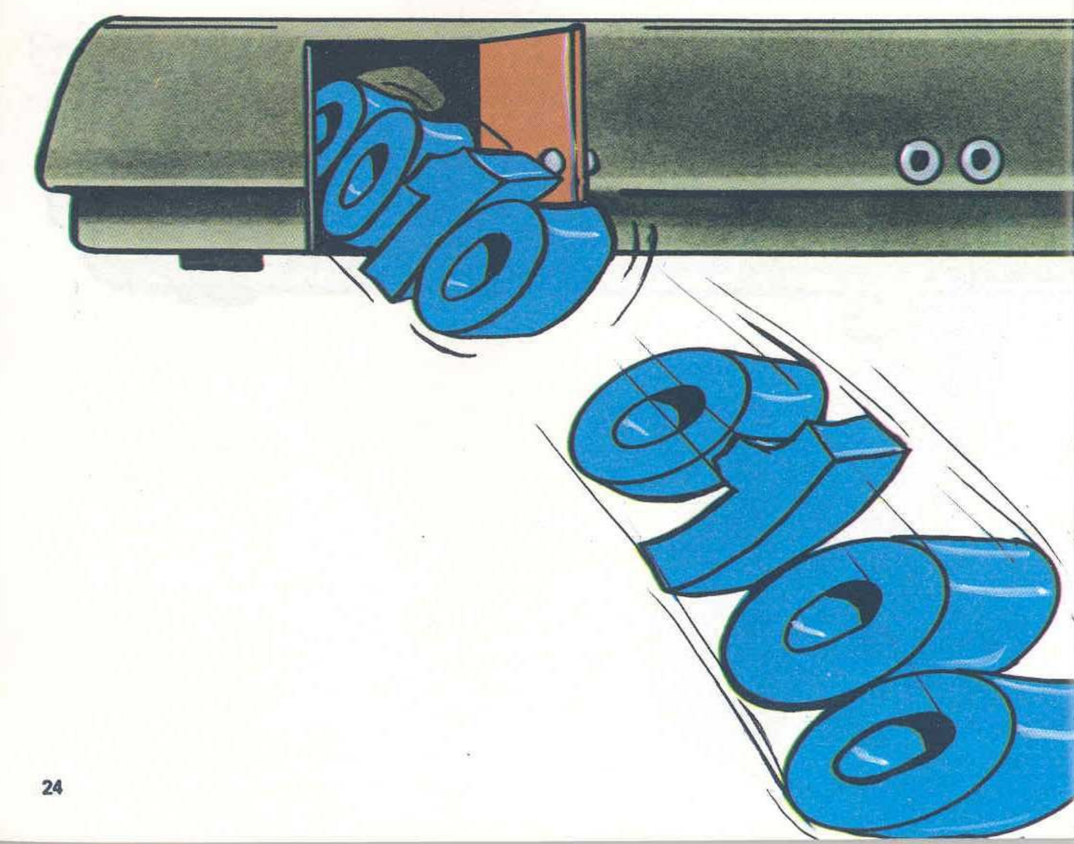
específicas de memoria. IN es la función correspondiente a PEEK:

```
IN dirección
```

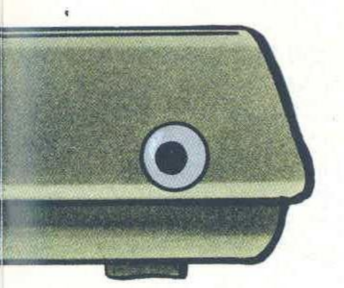
y tiene un único argumento (la dirección del port), proporcionando el valor numérico conectado a dicho port.
OUT, en cambio, es la instrucción correspondiente a POKE:

```
OUT dirección, valor
```

escribe el valor (numérico) en el port especificado por la dirección.
La principal diferencia entre PEEK/POKE e IN/OUT es que, mientras que todas las localizaciones de la memoria se comportan más o menos de la misma manera (algunas permiten únicamente la lectura, otras pueden ser leídas o escritas), el dispositivo que se maneja mediante la "dirección" puede actuar, dependiendo de su naturaleza, de muy variadas maneras.
Además, la instrucción OUT no implica ninguna memorización de datos.
Por lo tanto, es necesario conocer muy bien, además del port



conectado con el dispositivo, también el funcionamiento exacto de dicho dispositivo. Por ejemplo, existen algunas direcciones de entrada/salida que corresponden a ports de entrada/salida y que únicamente pueden recibir datos, como el altavoz: en este caso solamente tiene sentido utilizar OUT.



Otras direcciones corresponden a ports que sólo pueden proporcionar datos (por ejemplo, el teclado): en este segundo caso, solamente tiene sentido el uso de IN. Otras direcciones pueden recibir y proporcionar datos (por ejemplo, el port de conexión de la grabadora): aquí podremos utilizar tanto IN como OUT.
Los dispositivos de entrada/salida que has recibido con tu Spectrum son: el altavoz, el interface para la grabadora de casetes y el teclado.
Sin entrar demasiado en el tema, encontrarás a continuación dos brevísimos ejemplos. El primero, en base a la respuesta, modificará el color del borde de la pantalla:

```
10 INPUT "ESCRIBE UN NUMERO ENTRE 0 Y 7"; A
20 OUT 254,A
30 GO TO 10
```

El segundo, hará emitir un sonido por el altavoz:

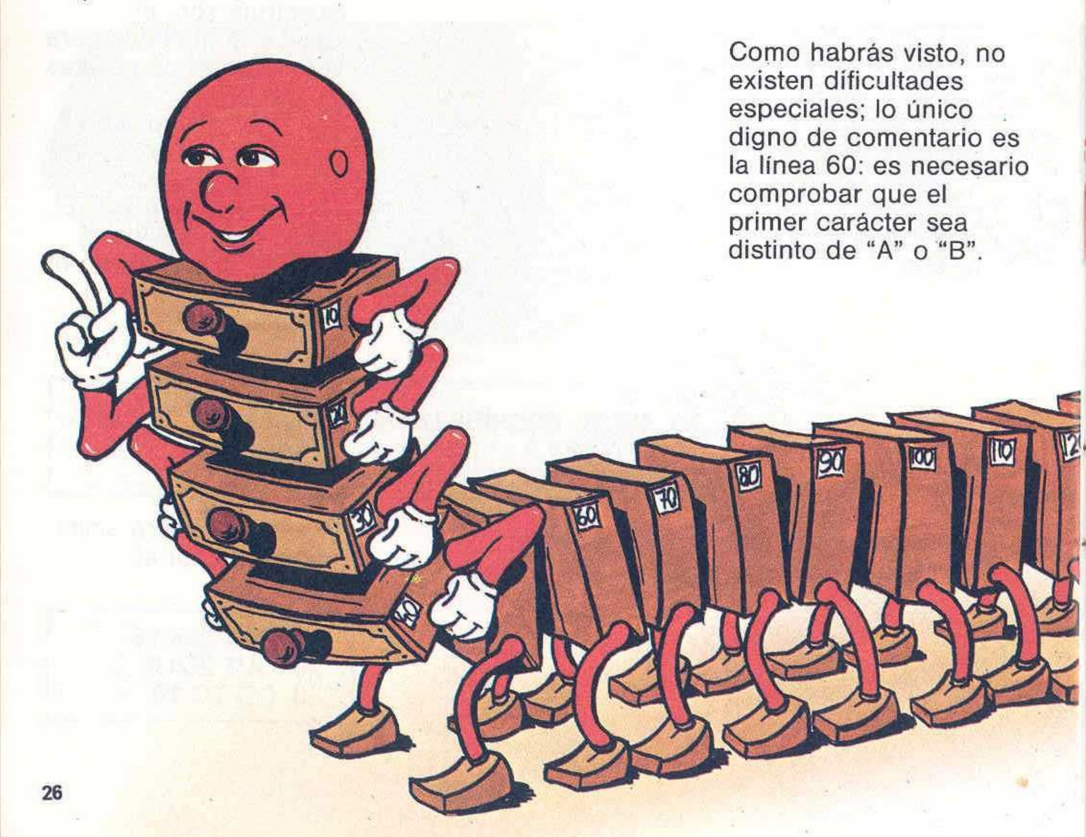
```
10 OUT 254,16
20 OUT 254,0
30 GO TO 10
```

Operaciones sobre cadenas

Los programas de hoy te enseñarán algunas posibles aplicaciones de las funciones para el tratamiento de cadenas. Veamos el primer ejemplo: imprimir una cadena, introducida por el teclado, únicamente si el primer carácter es distinto de "A" o "B".

```
 10 PRINT "TECLEA LA CADENA"
 20 INPUT N$
 30 REM A$ = PRIMERA LETRA DE LA CADENA
 40 LET A$ = N$ (1)
 50 REM COMPRUEBA SI EMPIEZA POR A O POR B
 60 IF (A$ = "A") OR (A$ = "B") THEN GOTO 100
 70 REM LA CADENA EMPIEZA CON UNA LETRA DISTINTA
 80 PRINT "LA CADENA ES: "; N$
 90 GOTO 110
100 PRINT "NO SE PUEDE IMPRIMIR LA CADENA PORQUE EMPIEZA
    POR:"; A$
110 STOP
```

Como habrás visto, no existen dificultades especiales; lo único digno de comentario es la línea 60: es necesario comprobar que el primer carácter sea distinto de "A" o "B".



Segundo ejemplo: escribir una palabra –introducida como de constumbre a través del teclado)– en sentido inverso, es decir, opuesto al normal (de derecha a izquierda).

```
10 PRINT "TECLEA LA CADENA
20 INPUT A$
30 LET K = LEN (A$)
40 FOR I = K TO 1 STEP –1
50 PRINT A$ (I);
60 NEXT I
70 STOP
```

También en este caso, el método es muy sencillo: se toma un carácter a la vez, de la palabra o frase a escribir, partiendo desde la izquierda y repitiendo la operación tantas veces como caracteres constituyan la propia palabra. Al final, en pantalla encontraremos visualizada la cadena original escrita al revés. Si aún tienes alguna duda, al añadir la línea

```
35 FOR J = 0 TO 500: NEXT J
```

introducimos un ciclo de retardo, que te ofrecerá la posibilidad de observar mucho más detalladamente la formación, carácter por carácter, de la palabra completa.
Otra posible solución hubiera podido ser:

```
10 PRINT " TECLEA LA CADENA"
20 INPUT A$
30 LET B$ = " "
40 LET K = LEN (A$)
50 FOR I = 1 TO K
60 LET A$ = A$ (1 TO K – I + 1)
70 LET C$ = A$ (LEN (A$))
80 LET B$ = B$ + C$
90 NEXT I
100 PRINT B$
110 STOP
```

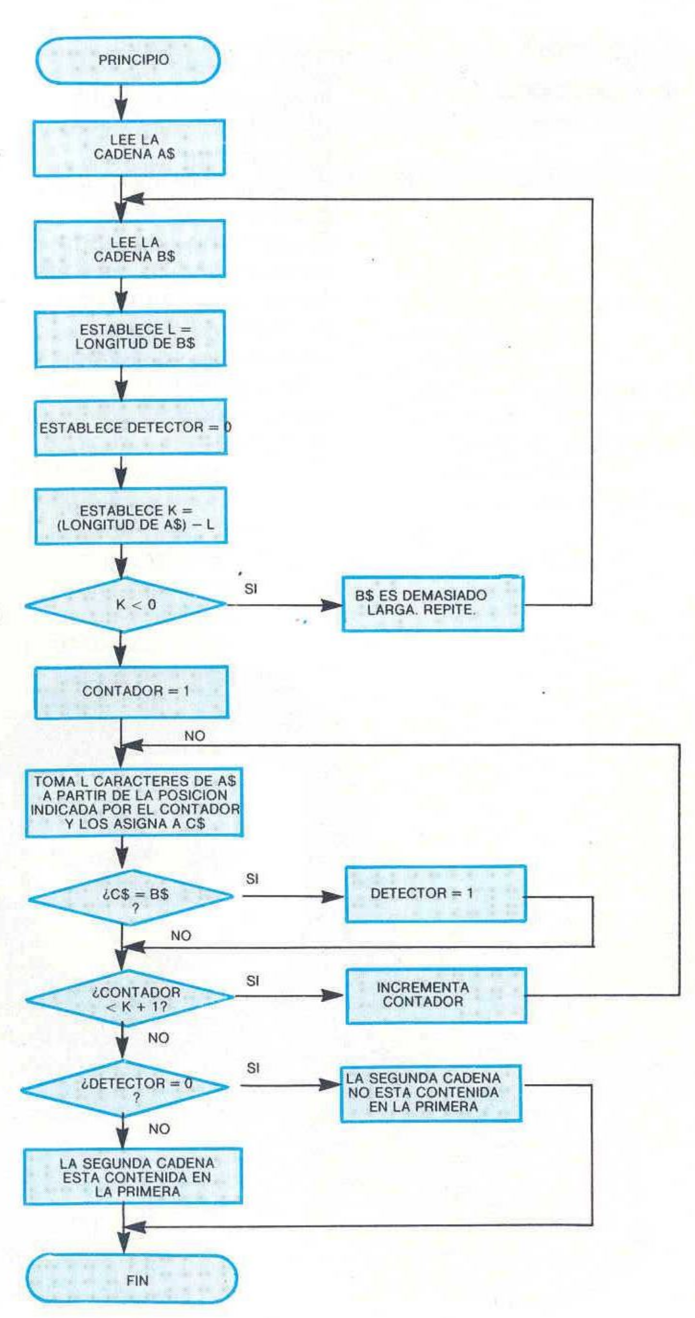
En este ejemplo, los distintos caracteres, en lugar de ser enviados directamente a la pantalla, se van añadiendo uno tras otro a la variable B$.
Una vez completado el ciclo (líneas 30-70) B$ contendrá la palabra invertida lista para imprimir.

Como último ejemplo, queremos resolver este problema: averiguar si una cadena está contenida en otra cadena (por ejemplo, la palabra "carga" está contenida en "descargadores", mientras que "manzana" no tiene nada que ver con "perro").
En primer lugar veremos esta posible solución:

```
 10 INPUT "TECLEA LA PRIMERA CADENA" ' A$
 20 INPUT "TECLEAR LA SEGUNDA CADENA" ' B$
 30 LET L = LEN (B$)
 40 LET DETECTOR = 0
 50 LET K = LEN (A$) − L
 60 IF K < 0 THEN PRINT "¡DEMASIADO LARGA! REPITE": GO TO 20
 70 FOR I = 1 TO K + 1
 80 LET C$ = A$ (I TO I + L − 1)
 90 IF B$ = C$ THEN LET DETECTOR = 1
100 NEXT I
110 IF DETECTOR = 0 THEN PRINT "LA SEGUNDA CADENA NO ESTA
    CONTENIDA EN LA PRIMERA."
120 IF DETECTOR = 1 THEN PRINT "LA SEGUNDA CADENA ESTA
    CONTENIDA EN LA PRIMERA."
130 STOP
```

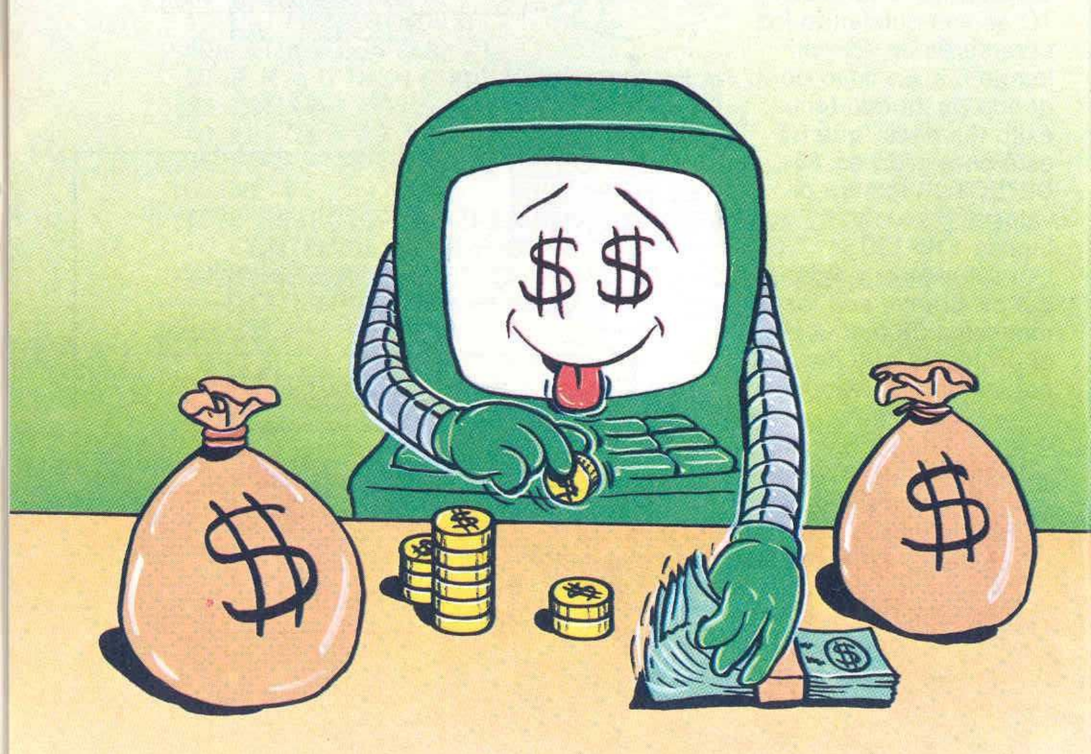
Haremos algunos comentarios.
Líneas 10-20: se piden como entradas las dos cadenas, asignando sus valores a las variables A$ y B$.
Líneas 30-40: la variable numérica L toma el valor dado por el número de caracteres

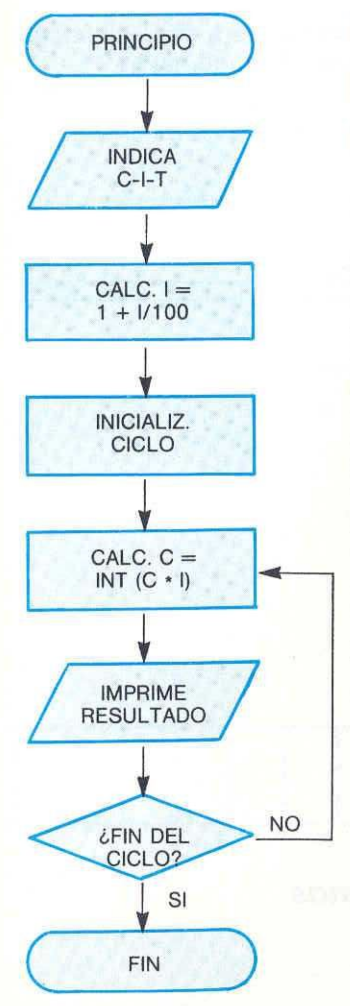
que componen la variable B$.
DETECTOR, en cambio, es una variable que únicamente podrá tomar dos valores, 0 y 1; valdrá 0 si A$ no contiene a B$, y 1 en el caso contrario.
Líneas 50-60: si la longitud de B$ resulta superior a la de A$, está claro que la comparación no es posible; será necesario teclear de nuevo B$.
Líneas 70-100: este es el corazón del programa.
Empleando la función TO se van cotejando los caracteres de A$ con los de B$. En caso de que la búsqueda tenga éxito (es decir, que B$ esté contenido en A$), DETECTOR tomará el valor 1.
Líneas 110-120: Constituyen la salida del programa, son los mensajes de respuesta al problema que nos habíamos planteado.



Capitales e intereses

El próximo listado es un programa de tipo financiero: el cálculo del interés compuesto y la impresión de su correspondiente tabla, ordenado por años.
El algoritmo empleado es muy sencillo, se trata fundamentalmente de:
A) Pedir los datos necesarios, es decir:
1. El capital (C) sobre el que efectuar el cálculo.
2. El interés (I).
3. El tiempo (T).
B) Calcular el total (capital + interés) de cada año, teniendo en cuenta que se considera como capital para el año siguiente el montante del anterior.
C) Imprimir la relación de los distintos años, poniendo cuidado en que los importes queden bien encolumnados.






```
10 INPUT "C"; C' "I"; I' "T"; T
20 LET I = 1 + I/100
30 FOR A = 1 TO T
40 LET C = INT (C * I)
50 PRINT A; TAB 31 – LEN STR$ C; C
60 NEXT A
```

Debido a la exigencia comentada en el apartado C, lo más cómodo será emplear las funciones de cadena (STR$) y (LEN) en conjunción con (TAB). Puesto que en tus futuros programas te encontrarás muchas veces con la necesidad de un encolumnado riguroso de los datos, será conveniente que te familiarices desde ahora con este tipo de instrucciones.

Comentarios al listado

10 INPUT de los valores de CAPITAL, INTERES y TIEMPO (en número de años).
20 Transformación de la tasa de interés, de porcentaje a decimal.
30 Establecimiento del ciclo en función de los años.
40 Cálculo del montante anual.
50 Impresión de la tabla con número del año y valor del montante.
Observa en especial la instrucción TAB (31 – LEN STR$ C) que tiene la tarea de situar correctamente el dato a imprimir.
60 Cierre del ciclo.

EJERCICIOS

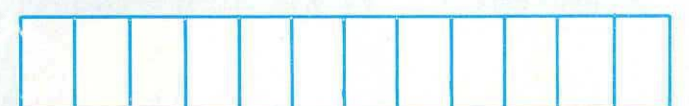
Anota en los espacios en blanco el resultado que preveas para cada ejercicio y después compruébalo con la solución de tu ordenador. Aunque hayas cometido un solo error deberás repasar la lección.

```
10 LET A$ = "12345"
20 LET B$ = "67"
30 LET C$ = "890"
40 PRINT LEN B$
50 PRINT LEN (A$ + B$ + C$)
60 PRINT LEN C$ – LEN B$
```

Para remarcar

```
10 LET T$ = "TITULO"
20 FOR C = 1 TO LEN T$
30 PRINT T$ (C); " ";
40 NEXT C
```

Antes de poner en marcha el programa escribe su salida.



Adivina donde se imprimirán las cadenas

```
10 CLS
20 INPUT "CADENA ="; S$
30 FOR I = 1 TO 10: PRINT : NEXT I
40 PRINT TAB (31 – LEN S$)/2; S$
50 PRINT AT 18,0; "¿OTRA VEZ?"
60 LET A$ = INKEY$: IF A$ = " " THEN GOTO 60
70 IF A$ = "s" THEN RUN
80 STOP
```

¿En qué línea?

¿Dónde, con respecto a las columnas?

SEIKOSHA SP-800

El fruto de la Investigación



La nueva impresora de **SEIKOSHA SP-800**, con un ordenador personal puede escribir ***96 combinaciones de letra diferentes***, desde 96 caracteres por segundo a 20 con muy alta calidad de letra, además es gráfica en alta densidad.

Su precio es de 69.900 Pts con ***introductor automático hoja a hoja***.

Con un pequeño ordenador personal, un procesador de textos puede costar alrededor de cien mil pesetas.

Infórmese y comprenderá por qué **las máquinas de escribir** tienen demasiados años.

Nuestra calidad es "SEIKO";
nuestros precios, únicos
Si desea más información,
consulte con nuestro distribuidor
más cercano, llame o escriba a:

DiRAC

DIRECCION COMERCIAL:
Av. Blasco Ibañez, 114-116
46022 VALENCIA
Tel (96) 372 88 85
Télex 62220
DIRECCION COMERCIAL EN CATALUÑA:
C/Muntaner, 60-2-4Pta
08011 BARCELONA
Tel (93) 323 32 19

ESTOS SON NUESTROS MODELOS:

MODELO	VELOCIDAD	COLUMNAS	TIPOS DE LETRA	P.V.P.R. INTERFACE PARALELO
GP-50S LA DEL SPECTRUM	40 cps	32	-	19.900
GP-50 LA PEQUEÑA	40 cps	46	2	25.900
GP-500 LA ECONOMICA	50 cps	80	2	47.900
GP-700 LA DE COLOR	50 cps	80-106	3	69.900
SP-800 LA PERFECCION	96 cps	80-137	20	69.900
BP-5200 LA DE OFICINA	200 cps	136-272	18	199.900
BP-5420 LA MAS RAPIDA	420 cps	136-272	18	299.900

* Los precios indicados son los recomendados para conexion tipo paralelo Centronics, para otro tipo de conexion, sufren un ligero incremento.

Este pie de página ha sido realizado integramente con la nueva impresora:

SEIKOSHA SP-800